Capítulo 1 Inicio de sesión

Conceptos clave

- En Linux, hay que comenzar las sesiones de usuario "iniciando la sesión" en la máquina.
- Para iniciar la sesión se debe contar de antemano con un nombre de usuario y una contraseña.
- Si una máquina Linux se encuentra ejecutando en un servidor X, los usuarios pueden iniciar la sesión utilizando la interfaz gráfica.
- Los usuarios pueden iniciar la sesión en cualquiera de las seis consolas virtuales.
- Las teclas **CTRL-ALT-F1** hasta **CTRL-ALT-F6** realizan el cambio entre las seis consolas virtuales.
- Las teclas **CTRL-ALT-F7** cambian al servidor X predeterminado.
- Si una máquina Linux lo soporta, los usuarios pueden iniciar la sesión en la red utilizando una shell remota.
- El comando **who** enumera los usuarios que actualmente tienen una sesión activa y la manera en que iniciaron la sesión.

Inicio de Sesiones Linux

El sistema operativo Unix, desarrollado originalmente a principios de los años setenta, precede los días de los computadores personales. Originalmente, muchos usuarios utilizarían Unix de manera simultánea operando en un computador central. En vez de que cada usuario tuviese un computador personal, utilizaban lo que se conoce comúnmente (hoy en día) como un"terminal tonto" - apenas un teclado y un monitor de texto conectados al computador central con una conexión serial. Para identificarse con el sistema operativo Unix, el usuario tenía que "iniciar sesión" primero con un userid y su contraseña.

Aunque Linux (y otras versiones de Unix) han crecido para tomar ventaja de los computadores personales al alcance de todos, no ha perdido los conceptos esenciales como los usuarios, las contraseñas, el inicio de sesión y las terminales basadas en texto. Esta lección describirá muchas de las formas para iniciar una sesión interactiva con un sistema Linux.

Consolas virtuales

Red Hat Enterprise Linux proporciona seis "consolas virtuales" disponibles para los usuarios que se encuentran físicamente frente al computador. Las consolas virtuales actúan como "terminales tontos". Para acceder a ellas se deben pulsar las teclas desde **CTRL-ALT-F1** hasta **CTRL-ALT-F6** con una consola virtual asociada a cada una de las primeras teclas de función (si se encuentra en un entorno gráfico debe recordar que **CTRL-ALT-F7**, le llevará de regreso a esta consola antes de intentar cambiar de consola virtual).

Al iniciar una sesión en una máquina Linux utilizando una consola virtual, la pantalla se verá así:

```
Red Hat Enterprise Linux ES release 4 (Nahant)
Kernel 2.6.9-5.EL on an i686

station login:
```

Para iniciar la sesión, el usuario digita su nombre de usuario sin espacios y pulsa la tecla **INTRO**. Después se le pide que escriba la contraseña , la cual no aparece en la pantalla, y luego pulsa**INTRO**. Iniciada la sesión, se recibe al usuario con un intérprete de comandos de la shell.

```
Red Hat Enterprise Linux ES release 4 (Nahant)
Kernel 2.6.9-5.EL on an i686

station login: elvis
Password:
[elvis@station elvis]$
```

La siguiente parte de este curso tratará de lo que se puede hacer desde el intérprete de comandos o "prompt" . Por ahora sólo vamos a aprender a terminar una sesión escribiendo el comando "**exit**" y pulsando**RETURN**. La consola virtual debe regresar a la pantalla original de inicio de sesión.

¿Por qué los usuarios quieren utilizar una consola virtual en vez del entorno gráfico? Primero, puede que la máquina no tenga suficiente memoria o hardware para soportar el entorno gráfico y puede que todo lo que se encuentre disponible sean las consolas virtuales. Sin embargo, incluso en las máquinas modernas, es muy frecuente que las consolas virtuales sean una interfaz más rápida y más eficaz para los usuarios experimentados. Como son implementadas directamente por el kernel de Linux, también son útiles en la reparación de sistemas en donde no estén funcionando los gráficos. Por último, puesto que los servidores de red usualmente pasan la vida encerrados en armarios, los administradores quieren a menudo evitar la complejidad que la interfaz gráfica le añade al sistema.

El Entorno Gráfico X

Cuando Linux opera en computadores personales modernos, utiliza un programa de bajo nivel conocido como el "servidor X" para proveer el entorno gráfico. Como usuario del sistema, usted no interactúa directamente con el servidor X, sino que proporciona la base para todos los programas gráficos que ejecuta.

Cuando el servidor X inicia, como parte de la secuencia de inicio del sistema, generalmente busca la "primera" consola virtual disponible. Debido a que hay seis consolas virtuales para los terminales de texto, el servidor X usualmente toma la séptima. Si se encuentra en una consola virtual y piensa que hay un servidor X ejecutando en la máquina, usted puede cambiarse a ésta pulsando las teclas **CTRL-ALT-F7**.

Si un sistema se encuentra configurado para arrancar en el entorno gráfico, entonces se le presentará al usuario el Administrador de Inicio de Sesión, el cual se ve de la siguiente manera:

Figure 1. Administrador de Inicio de Sesión de Red Hat Enterprise Linux

De nuevo, un usuario inicia la sesión al escribir su nombre de usuario seguido de **INTRO** y su contraseña seguida de **INTRO**.

Terminales en el entorno gráfico

El menú de aplicaciones en el entorno gráfico se encuentra en la esquina superior izquierda y sirve como portal de entrada a una amplia variedad de aplicaciones gráficas, incluyendo navegadores, editores de texto, editores de imágenes, juegos y mucho más. Aunque le alentamos a que explore estas aplicaciones , comenzaremos este curso enfocándonos en los fundamentos de operación dentro del entorno Linux tales como la administración de archivos y procesos. Por ahora, la aplicación más importante para usted en el entorno gráfico es la terminal.

En Red Hat Enterprise Linux, la terminal comúnmente utilizada es la llamada **gnome-terminal**. Se puede abrir una nueva ventana **gnome-terminal** haciendo click en el botón derecho del ratón sobre el fondo del escritorio y seleccionando la primera opción del menú "Abrir una terminal".

Figure 1. Apertura de una gnome-terminal

Los usuarios experimentados de Linux con frecuencia prefieren la versatilidad y la potencia de la interfaz de la línea de comandos en vez de las aplicaciones gráficas. De manera similar a una consola virtual, el terminal proporciona una interfaz de línea de comandos de shell. Debido a que el usuario ya ha iniciado sesión para poder arrancar en la sesión gráfica, no necesita iniciar otra vez la sesión al abrir una nueva terminal.

Figure 2. gnome-terminal

Cuando termine puede cerrar la **gnome-terminal** escribiendo el comando **exit** (seguido de **INTRO**), o haciendo click en el botón izquierdo de la "opción para cerrar" que se encuentra en la esquina superior derecha de la terminal.

Salida del entorno gráfico

Cuando un usuario termina con el entorno gráfico, este puede salir seleccionando la última opción del menú de Acciones que está justo a la derecha del menú de Aplicaciones. Esto cerrará todas las ventanas y regresará el entorno gráfico a su pantalla de inicio original, listo para el siguiente usuario.

Figure 1. Salida del entorno gráfico

Uso del comando who para determinar quién se encuentra en el sistema

Los usuarios pueden utilizar el comando **who** para determinar quién se encuentra en el sistema y cómo iniciaron la sesión. Desde cualquier terminal los usuarios pueden escribir **who** y presionar la tecla **INTRO**. Deben ver una salida similar a la siguiente:

```
[elvis@station elvis]$ who
elvis    tty2         May  5 15:07
root     tty1         May  3 07:50
blondie  :0           May  5 08:48
blondie  pts/0        May  5 09:03 (:0.0)
```

La primera columna lista los usuarios que han iniciado sesión y las últimas columnas establecen la hora en que la iniciaron. Con un poco de experiencia, la segunda columna le dice el lugar en donde el usuario inició la sesión. Para los primeros dos usuarios, elvis y root, `tty1` y `tty2` se refiere a las consolas virtuales 1 y 2 respectivamente. La primera entrada para blondie, `:0`, se refiere al servidor X mismo y `pts/0` se refiere a la primera terminal abierta en el servidor X. Sin detenernos mucho en los detalles, podemos ver que elvis ha iniciado sesión en la segunda consola virtual, root en la primera y blondie ha iniciado sesión utilizando el entorno gráfico.

Inicio de sesión en una máquina a través de la red

Al utilizar la opción **ssh** ("secure shell"), los usuarios pueden iniciar una sesión fácilmente en máquinas remotas ya sea en el mismo lugar o al otro lado del mundo. Asumiendo que la máquina remota se encuentra configurada para permitir el inicio de sesión de manera remota y que el usuario tiene una cuenta en la máquina, el iniciar una sesión en una máquina remota puede ser tan fácil como iniciar una sesión en una consola virtual. En el siguiente ejemplo, elvis iniciará una sesión en una consola virtual en la máquina local llamada *station.redhat.com*. Después utilizará **ssh** para iniciar la sesión en una máquina remota *nimbus.example.com*, muy posiblemente al otro lado del mundo.

Preste mucha atención al intérprete de comandos en el siguiente ejemplo. Como con frecuencia los usuarios de Linux van "saltando" de máquina en máquina utilizando shells remotas, el intérprete de comandos ha sido diseñado para ayudar al usuario a recordar en qué máquina está operando la shell.

```
Red Hat Enterprise Linux ES release 4 (Nahant) (Taroon)
Kernel 2.4.21-4.0.1-EL on an i686
```

```
station login: elvis
Password:
Last login: Thu Apr  3 13:03:06 from hedwig
[elvis@station elvis]$ who
elvis    tty2         May  3 07:48
[elvis@station elvis]$ ssh elvis@nimbus.example.com
elvis@nimbus.example.com's password:
Last login: Thu May  1 17:38:43 2003 from station.redhat.com
[elvis@nimbus elvis]$ who
elvis    pts/1        May  3 11:59 (station.redhat.com)

[elvis@nimbus elvis]$ exit
Connection to nimbus.example.com closed.
[elvis@station elvis]$ exit
```

Observe que cuando elvis ejecutó el comando **who** en la máquina remota, ésta reportó la máquina desde donde inició la sesión llamada *station.redhat.com*.

Hay clientes ssh para sistemas operativos diferentes de Linux. Por ejemplo, la aplicación Open Source PuTTY se puede instalar en casi cualquier máquina windows y se puede utilizar para abrir un intérprete de comandos en un servidor Linux remoto.

Ejemplos

Uso de consolas virtuales

En el siguiente ejemplo, un usuario inicia sesión en la primera consola virtual como el usuario elvis, pero luego se da cuenta de que su sistema se encuentra mal configurado. Para poder arreglar esto, va a cambiar a la segunda consola virtual e iniciará sesión como root y arreglará el problema. Luego saldrá y regresará a la primera consola virtual listo para iniciar.

```
Red Hat Enterprise Linux ES release 4 (Nahant)
Kernel 2.6.9-5.EL on an i686

station login: elvis
Password:
Last login: Mon May  5 15:07:16 on tty2
[elvis@station elvis]$ cat /tmp/README
cat: /tmp/README: Permission denied
[elvis@station elvis]$ who
elvis    tty1         May  5 15:54
```

En este momento, elvis ha tratado de leer el archivo `/tmp/README`, pero los permisos del archivo no le permitieron hacerlo. Debido a que el usuario conoce la contraseña para la cuenta de root (del administrador) en la máquina, puede arreglar el problema cambiándose a otra consola virtual (utilizando las teclas **CTRL-ALT-F2**), iniciando sesión como root y cambiando los permisos del archivo.

```
Red Hat Enterprise Linux ES release 4 (Nahant) (Taroon)
Kernel 2.4.21-4.0.1-EL on an i686
```

```
station login: root
Password:
Last login: Mon May  5 15:54:18 on tty6
[root@station root]# who
root     tty2         May  5 15:55
elvis    tty1         May  5 15:54
[root@station root]# chmod go+r /tmp/README
[root@station root]# exit
```

Observe que la salida del comando **who** muestra que el usuario inició sesión como elvis en la primera consola virtual e inició sesión como root en la segunda. El usuario ahora cambia de vuelta a la primera consola virtual utilizando **CTRL-ALT-F1** y reanuda la sesión.

```
[elvis@station elvis]$ cat /tmp/README

Kudzu FAQ:
----------

1) So, what is kudzu?

kudzu is the hardware autodetection & configuration tool, originally
introduced in Red Hat Linux 6.1. It detects changes in the system's
hardware configuration, and gives the option of configuring
...
```

Ejercicios en línea

Uso de múltiples consolas virtuales

Lab Exercise

Objetivo: Aprender a administrar múltiples consolas virtuales.

Estimated Time: 10 mins.

Especificaciones

Su máquina debe haber sido configurada con múltiples cuentas, todas con la misma contraseña. Si por ejemplo su nombre de usuario es `elvis`, también debe tener cuentas llamadas `elvis_a`, `elvis_b` y `elvis_c`,todas con la misma contraseña.

Para este ejercicio debe iniciar sesión en las primeras cuatro consolas virtuales, utilizando cada una de sus cuentas en orden.

Consola Virtual	Nombre de usuario
1	*username*
2	*username*_a
3	*username*_b
4	*username*_c

Si completó el ejercicio correctamente, debería poder ejecutar el comando **who** desde cualquier terminal y obtener una salida similar a la siguiente:

```
[elvis@staion elvis]$ who
elvis_a  tty2         May  5 16:18
elvis    tty1         May  5 16:18
elvis_b  tty3         May  5 16:18
elvis_c  tty4         May  5 16:18
```

Deliverables

Question 1

1. Cuatro sesiones activas en cada una de las 4 consolas virtuales

Capítulo 2 El kernel, los programas y los procesos

Conceptos clave

- El kernel de Linux es un sistema operativo que proporciona un entorno para la ejecución de procesos.
- Los programas son archivos ejecutables (usualmente binarios) que se encuentran en un sistema de archivos.
- Un proceso es una instancia de un programa en ejecución.
- Una shell es un proceso interactivo que permite al usuario specificar otros procesos a ejecutar.
- Aunque Red Hat Enterprise Linux se entrega con diferentes versiones de shells, la shell predeterminada es un programa llamado **bash**.
- El comando **ps** lista los procesos iniciados desde una terminal individual. El comando **ps aux** le da una lista detallada de todos los procesos operando actualmente en una máquina.

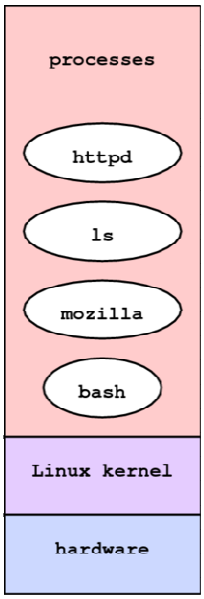
Discussion

El kernel de Linux

El kernel de Linux es un sistema operativo así como Windows, MacOS y OS/2. El sistema operativo es un entorno en el cual se ejecuta un programa. proporcionándole los mecanismos a los programas para interactuar con el hardware del computador. El kernel le permite saber a un programa cuándo se pulsa una tecla, cuándo se mueve el ratón o cuándo llegó un paquete destinado para el programa a través de la red . El kernel le permite a los programas acceder al disco duro, quizás para leer los archivos de configuración o almacenar datos.

Así como sería difícil explicar lo que es un lenguaje a una persona que ha crecido con el conocimiento de un solo idioma, con frecuencia es difícil explicar lo que es un sistema operativo a aquellas personas nuevas en el área de los computadores o a aquellos que solamente han utilizado un sistema operativo. El sistema operativo funciona en un nivel tan bajo que los usuarios no interactúan con éste directamente y a menudo no se dan cuenta de que los programas que se encuentran ejecutando lo están utilizando.

Figure 1. El kernel y los procesos



Los programas

Los programas son archivos almacenados en el disco duro de un computador. Un programa es una serie de instrucciones básicas que le indican al sistema operativo qué hacer y cuándo hacerlo de una manera bastante detallada y específica. Las siguientes pueden ser las instrucciones escritas en el archivo **/usr/X11R6/bin/xcalc**, un programa común de calculadora.

"Mientras que el usuario no esté haciendo nada, el cursor titila. Cuando el usuario pulsa una tecla, si es un número, lo escribe en la pantalla. Si no es un número lo ignora, a menos que sea la tecla intro. Si es la tecla intro, añade el número anterior al total y muestra el total."

Claro está que las instrucciones no se encuentran en inglés ni en ningún otro lenguaje humano. En cambio, un computador en el nivel más bajo sólo sabe cómo realizar un pequeño número de tareas, digamos 256 de ellas. Imagine que cada una de esas tareas estuviesen numeradas de 0 a 255. Los programas se ven en realidad más como las siguientes instrucciones:

"Realice la tarea 23. Si el resultado es mayor que 0, entonces haga la tarea 45. De otra forma, complete la tarea 82. Tome el resultado y almacénelo en la memoria."

Debido a que las tareas que un computador puede llevar a cabo varían dependiendo del tipo de CPU, y que diferentes sistemas operativos podrían listar las tareas de forma diferente, los programas compilados para un tipo de sistema operativo, por lo general, no ejecutarán en otros sistemas operativos.

Los procesos

Cuando un usuario le pide al kernel que ejecute un programa, el kernel lee las instrucciones que se encuentran en el el archivo del programa y las carga en la memoria. Después empieza a llevar a cabo las instrucciones . La copia del programa ejecutando en la memoria se denomina un proceso. Observe que el mismo programa se puede cargar en la memoria y se puede ejecutar más de una vez para que en cualquier momento el kernel pueda ejecutar varios procesos del mismo programa.

Linux, como la mayoría de sistemas operativos modernos, es un sistema operativo de funciones múltiples. Esto significa que el kernel parece estar ejecutando múltiples procesos de manera simultánea. En realidad, el kernel se encuentra ejecutando un sólo proceso por un breve periodo de tiempo, usualmente alrededor de 50 milisegundos. Luego intercambia ese proceso por otro con sus 50 milisegundos de acción. Finalmente, el kernel termina todos los procesos y comienza de nuevo. El rápido cambio de proceso a proceso le da la impresión a los usuarios de la máquina de que todos los procesos se están ejecutando simultáneamente.

Las shells

Los usuarios de Linux con frecuencia utilizan un tipo especial de programa llamado shell para interactuar con el kernel. La shell ejecuta dentro de una terminal emitiendo un "prompt"y esperando a que le digan qué hacer. Luego, el usuario le pide a la shell que ejecute un programa escribiendo el nombre del programa. En el siguiente ejemplo, el usuario elvis le pide a la shell que ejecute un programa sencillo de calendario llamado **cal**. La shell ejecuta obedientemente el programa y reporta la salida del programa a la terminal:

```
[elvis@station elvis]$ cal
      May 2003
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31

[elvis@station elvis]$
```

Debido a que los usuarios utilizan una shell para llevar a cabo casi todas las tareas en el sistema, la shell es un programa importante y sofisticado. Gran parte de este curso se dedicará a aprender a utilizar la shell para realizar tareas de manera efectiva.

Cuando el usuario le pide a la shell que ejecute un programa, se dice que el usuario está especificando un comando para la shell. Con frecuencia la shell se conoce como la interfaz de línea de comandos.

El comando ps

El comando **ps** se utiliza comúnmente para enumerar los procesos ejecutándose en un sistema. El comando **ps**, sin ningún argumento, muestra todos los procesos que se iniciaron desde una sola terminal (o una terminal en un entorno gráfico). En el siguiente ejemplo, elvis descubre que actualmente su terminal tiene dos procesos en ejecución: su shell **bash** y el comando mismo **ps**.

```
[elvis@station elvis]$ ps
  PID TTY          TIME CMD
 1378 pts/1    00:00:00 bash
 1418 pts/1    00:00:00 ps
```

La primera columna muestra el id del proceso (PID del inglés Process ID). Todo proceso ejecutándose en el sistema se identifica con un ID de proceso único. La segunda columna identifica la terminal de elvis y la tercera reporta el tiempo de CPU que el proceso ha utilizado.

El comando **ps aux** muestra una lista detallada de todos los procesos ejecutándose en el sistema. Por ahora no explicaremos lo que significa "aux", sólo piense que es como un acto de magia negra que debe memorizar. Tampoco le explicaremos todos los campos, todo esto se discutirá en el siguiente cuaderno. Simplemente observe que en cualquier momento hay, por lo general, un gran número de procesos ejecutándose en un sistema Linux (note que en la siguiente salida se han eliminado bastantes líneas y se han reemplazado con "..."):

```
[elvis@station elvis]$ ps aux
USER       PID %CPU %MEM    VSZ  RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1  1388  460 ?        S    18:52   0:04 init [
root           2   0.0   0.0      0     0 ?          SW    18:52    0:00
[keventd]
...
root       548  0.0  0.2  1460  580 ?        S    18:53   0:00 syslogd
-m 0
root       552  0.0  0.1  1384  428 ?        S    18:53   0:00 klogd -
x
rpc        570  0.0  0.2  1560  552 ?        S    18:53   0:00 portmap
root         622   0.0   0.2   1496   632 ?          S     18:53    0:00
/sbin/cardmgr
root       628  0.0  0.0     0    0 ?        SW   18:53   0:00 [kapmd]
root         633   0.0   0.1   1380   484 ?          S     18:53    0:00
/usr/sbin/apmd -p
root         644   0.0   0.5   3524  1512 ?          S     18:53    0:00
/usr/sbin/sshd
root       655  0.0  0.3  2040  860 ?        S    18:53   0:00 xinetd
-stayalive
root       674  0.0  0.9  6204 2504 ?        S    18:53   0:00
sendmail: accepti
```

```
smmsp      683  0.0  0.8  6004 2256 ?        S    18:53   0:00
sendmail: Queue r
root       693  0.0  0.1  1424  444 ?        S    18:53   0:00 gpm -t
ps/2 -m /d
root       702  0.0  0.2  1580  672 ?        S    18:53   0:00 crond
xfs        781  0.0  1.5  5272 4004 ?        S    18:53   0:00 xfs -
droppriv -da
root       800  0.0  0.2  3416  544 ?        S    18:53   0:00 rhnsd -
-interval
root       807  0.0  0.1  1364  396 tty1     S    18:53   0:00
/sbin/mingetty tt
...
root      1375  0.0  0.3  4120  968 pts/1    S    20:11   0:00 su -
elvis
elvis     1378  0.0  0.5  4324 1404 pts/1    S    20:11   0:00 -bash
elvis     1438  0.0  0.2  2648  696 pts/1    R    20:30   0:00 ps aux
```

Capítulo 3 El sistema de archivos

Conceptos clave

- La base del árbol de directorios de Linux es un directorio llamado `/` y conocido como el" directorio root".
- Las referencias *absolutas* de archivos, tales como `/home/elvis/lsout.txt`, se refieren a los archivos relativos al directorio raíz.
- Las referencias relativas de archivo, *tales como `lsout.txt`, hacen referencia a archivos relativos al directorio de trabajo actual de un proceso.*
- El comando **ls** se utiliza para enumerar el contenido de un directorio.
- El comando **cat** se utiliza para examinar el contenido de un archivo.
- Al utilizar el caracter > desde la línea de comandos de la shell, la salida de un comando puede redireccionarse desde la terminal a un archivo.
- El directorio home es uno de los pocos lugares en donde los usuarios pueden crear archivos nuevos.

Dar nombres a archivos y directorios

En Linux,la información y los programas se encuentran almacenados en discos como archivos. Los archivos se encuentran agrupados en los directorios, los cuales pueden contener archivos y otros directorios (otros sistemas operativos con frecuencia se refieren a directorios como "carpetas"). Esta jerarquía de directorios que contienen directorios se conoce a menudo como el "árbol de directorios".

La base del árbol de directorios es un directorio llamado simplemente `/` y se conoce como "el directorio raíz". Los archivos dentro del directorio raíz se conocen como */nombre del archivo*. En Red Hat Enterprise Linux, el directorio raíz contiene principalmente otros directorios tales como `/etc`. Los archivos dentro de estos subdirectorios se conocen como `/etc`/*nombre del archivo* y el proceso continúa así para cada nuevo nivel de subdirectorios. Por ejemplo, el archivo `network` que se

encuentra en el directorio `sysconfig`, que a su vez se encuentra en el directorio `etc`, el cual está en el directorio root `/`, puede conocerse como `/etc/sysconfig/network`.

Obviamente, el iniciar en el directorio raíz cada vez que se refiera a un archivo es bastante trabajo. Afortunadamente, Linux proporciona una manera más fácil. Todo proceso, incluyendo la shell de un usuario, utiliza un "directorio actual de trabajo" por contexto. Los archivos en un directorio actual de trabajo se conocen simplemente como *nombre de archivo*, sin la barra oblicua inicial. Los archivos en los subdirectorios del directorio actual de trabajo se conocen como `nombre del directorio/`*nombre del arhivo*, de nuevo sin la barra oblicua inicial. Por ejemplo, si el directorio actual de trabajo de un proceso fuese `/etc`, el archivo `network` mencionado anteriormente se conocería como `sysconfig/network`. Si el directorio de trabajo fuese `/etc/sysconfig` entonces el archivo podría ser simplemente `network`.

En resumen, siempre hay dos maneras de referirse a un archivo. Las referencias de archivo relativas al directorio raíz, siempre comienzan con una barra oblicua inicial `/` y se llaman referencias absolutas *absolute*. Las referencias de un archivo relativas al directorio de trabajo actual comienzan con cualquier cosa que no sea una `/` y se conocen como referencias *relativas*.

Listado del contenido de un directorio con ls

Desde una shell los usuarios pueden utilizar el comando **ls** para hacer un listado del contenido de un directorio (imagínese que **ls** es la forma corta para la palabra "lista"). En el siguiente ejemplo, el usuario elvis quiere hacer una lista del contenido del directorio `/etc/sysconfig/rhn`.

```
[elvis@station elvis]$ ls /etc/sysconfig/rhn
rhn-applet  systemid  up2date-keyring.gpg  up2date-uuid
rhnsd       up2date   up2date.rpmnew
```

El comando **ls**, cuando se emite sin ningún argumento (por ejemplo, sin especificar un directorio) lista el contenido del directorio de trabajo actual de la shell. Si utiliza una terminal con colores,el comando **ls** también le pone color a los nombres de los directorios para ayudar a distinguir el contenido de los directorios que tienen archivos comunes (en blanco) y los que son directorios (en azul).

El **ls** es un comando muy flexible que puede proporcionar mucha información. Este se estudiará en más detalle en lecciones posteriores.

Ver el contenido de un archivo con cat

Mientras el comando **ls** lista los archivos que se encuentran en un directorio dado, éste no revela el contenido de los archivos. Aunque hay varios comandos disponibles para ver los archivos, el comando más sencillo es **cat**. Cuando se le da una lista de archivos al comando **cat** este con**cat**ena los archivos a la salida del terminal. Si sólo se le da el nombre de un archivo entonces mostrará el contenido de solo ese archivo.

En el siguiente ejemplo, el usuario elvis quiere ver el contenido del archivo de configuración `/etc/hosts`.

```
[elvis@station elvis]$ cat /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1       localhost.localdomain   localhost
192.168.0.254   server1.example.com  server1
192.168.0.1     station1.example.com station1
```

Por ahora, no se preocupe de lo que significa el contenido, sólo tenga en cuenta que el comando **cat** presenta todo el contenido de este archivo de cinco líneas.

Observe que si le pide a **cat** que presente un archivo muy largo o un archivo binario (no texto), **cat** lo hará obedientemente. Hay comandos más sofisticados para ver archivos largos, una pantalla a la vez, lo cual se verá más adelante.

Redireccionamiento de salidas de los comandos a archivos

Cuando los comandos **ls** y **cat** se ejecutan su salida se presenta en la terminal. En Linux, la mayoría de los comandos que generan texto en la salida utilizan un concepto común de Unix denominado "salida estándar". Por defecto se encuentra conectado a la terminal. La shell **bash** permite a los usuarios "redireccionar" la salida estándar a otros lugares. Por ahora, vamos a aprender el caso más fácil: la utilización del caracter > para redireccionar la salida estándar a un archivo.

En el siguiente ejemplo, el usuario elvis de nuevo va a hacer una lista del contenido del directorio `/etc/sysconfig/rhn`, pero quiere redireccionar la salida a un archivo recién creado.

```
[elvis@station elvis]$ ls /etc/sysconfig/rhn > lsout.txt
[elvis@station elvis]$ ls
lsout.txt
[elvis@station elvis]$ cat lsout.txt
rhn-applet
rhnsd
systemid
up2date
up2date-keyring.gpg
up2date.rpmnew
up2date-uuid
```

La salida del comando **ls /etc/sysconfig/rhn** no se mostró en la terminal sino que se introduce en el archivo recién creado `lsout.txt`. Despúes elvis toma el comando **ls** de su directorio de trabajo actual y ve el archivo recién creado. Después utiliza el comando **cat** para observar el contenido del archivo. En "lenguage Unix" elvis "redireccionó la salida del comando **ls** al archivo `lsout.txt`".

Permisos y el directorio de inicio de un usuario

Observe lo que pasa cuando elvis trata de redireccionar la salida a un archivo que se encuentra en un lugar diferente al directorio actual de trabajo de la shell.

```
[elvis@station elvis]$ ls /etc/sysconfig/rhn > /etc/lsout.txt
-bash: /etc/lsout.txt: No such file or directory
```

El usuario elvis se ha encontrado con otro concepto común en Linux: propiedad y permisos de un archivo. El usuario elvis trató de crear el nuevo archivo `/etc/lsout.txt`, pero elvis no tiene el permiso para crear archivos en el directorio `/etc`.

Por defecto, en Red Hat Enterprise Linux, los usuarios no pueden crear archivos en cualquier lugar. De hecho, sólo hay unos pocos lugares en donde se pueden crear archivos. Todo usuario tiene un directorio de inicio, en donde puede crear nuevos archivos (y nuevos subdirectorios). Afortunadamente, cuando los usuarios inician una sesión en Linux, su shell utiliza su directorio inicial como su directorio actual de trabajo. Por defecto en Red Hat Enterprise Linux, el directorio inicial de un usuario se llama `/home/`*`username`*, donde*`username`* se reempla con el nombre del nombre del usuario.

Otros cuaderno abordarán el sistema de archivos y los permisos en mucho más detalle. Por ahora, sólo tenga en cuenta que a los usuarios, generalmente, sólo se les permite crear archivos en su directorio de inicio.

Ejemplos

Redirección de la salida de un comando a un archivo

El usuario prince quiere utilizar el comando **cal** para almacenar un calendario del mes actual en el archivo `calendar.txt`.

```
[prince@station prince]$ cal                    ❶
      May 2003
Su Mo Tu We Th Fr Sa
                1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31

[prince@station prince]$ cal > calendar.txt     ❷
[prince@station prince]$ ls                     ❸
calendar.txt
[prince@station prince]$ cat calendar.txt       ❹
      May 2003
Su Mo Tu We Th Fr Sa
                1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

❶ prince primero ve la salida del comando **cal** directamente.

❷ prince ejecuta otra vez el comando **cal** redireccionando la salida al archivo `calendar.txt`

❸ prince confirma que el nuevo archivo ha sido creado listando el contenido de su directorio de trabajo actual (en este caso su directorio de inicio).

❹ prince examina el contenido del nuevo archivo para confirmar que contiene la salida del comando **cal**.

Ejercicios en línea

El comando ls

Lab Exercise
Objetivo: Redireccionar la salida del comando a un archivo recién creado.

Tiempo estimado: 5 minutos.

Especificaciones

Utilice el comando **ls** para hacer una lista del contenido del directorio `/etc`, redireccionando la salida del comando a un archivo llamado `lsetc.txt` en su directorio de inicio.

Deliverables

A title

Question 1

1. El archivo `lsetc.txt` en su directorio de inicio, el cual contiene la salida del comando **ls /etc**.

El siguiente comando proporciona una posible solución a este ejercicio.

```
[student@station student]$ ls /etc > lsetc.txt
```

Capítulo 4 Ejecución de comandos

Conceptos clave

- Como cualquier otro idioma la shell **bash** utiliza una gramática específica.
- La primera palabra en cualquier línea de comandos es el nombre del programa que se va a ejecutar.

- El comportamiento de los comandos se puede modificar con las opciones de la línea de comandos (usualmente opcional), las cuales siempre empiezan con uno o dos guiones (- o --).
- Las palabras u opciones que siguen después del comando se llaman argumentos del comando *arguments*
- Algunas opciones de la línea de comandos también toman argumentos.
- Los comandos usualmente soportan las opciones **--help**, **-h** o **-?**, las cuales presentan un resumen de cómo utilizar el comando.

Gramática de la línea de comandos

Aunque la interfaz de la línea de comandos tiene muchas ventajas, incluyendo la eficiencia y la flexibilidad, la simplicidad no es una de ellas. Afortunadamente, existen convenciones que (casi) todos los programas siguen. Aprender nuevos programas será mucho más fácil si toma un poco de tiempo para aprender estas convenciones.

Si los comandos que se introducen en la línea de comandos se comparan con las oraciones en inglés entonces los comandos tienen verbos, adverbios y objetos directos. El verbo es el comando a ejecutar, los adverbios son las diferentes opciones que se pueden utilizar para modificar el comportamiento del comando (“calmadamente” o “con fluidez”) y las palabras que quedan son los objetos directos (sobre lo que el comando debe actuar). Además, así como en los idiomas, hay irregularidades y para casi toda regla mencionada habrá excepciones.

Comandos

La primera palabra en cualquier comando es generalmente el nombre de un programa que vive como archivo en algún lugar del sistema de archivos. [1] Por ejemplo, la lección anterior utilizó los comandos **ps** y **who**. Si queremos saber qué archivo contiene estos programas, hay un programa llamado **which** que nos puede ayudar. El comando **which** seguido del nombre de otro comando le dirá "qué" archivo se está ejecutando.

```
[elvis@station elvis]$ which ps
/bin/ps
[elvis@station elvis]$ which who
/usr/bin/who
```

Cuando se ejecuta un comando, el proceso de la shell le pide al kernel que ejecute el programa especificado como un proceso separado y hace que la salida (o mejor, la salida estándar) del proceso se escriba en la terminal. Entonces la shell realiza una pausa hasta que el proceso del comando termina. Una vez el comando termina, la shell presenta otro intérprete de comandos y espera a que se le diga que hacer.

Opciones de la línea de comandos

Se puede modificar el comportamiento de muchos de los comandos especificando las diferentes opciones de la línea de comandos. Un ejemplo fácil es el comando **ls** utilizado para listar el contenido de un directorio. Considere los siguientes tres usos del comando **ls** cada vez que lista los archivos en el directorio `/usr`:

```
[elvis@station elvis]$ ls /usr
bin   etc    include   lib      local  share  tmp
dict  games  kerberos  libexec  sbin   src    X11R6
[elvis@station elvis]$ ls -s /usr
total 132
  40 bin      4 games       40 lib         8 sbin      0 tmp
   4 dict     8 include      4 libexec     8 share     4 X11R6
   4 etc      4 kerberos     0 local       4 src
[elvis@station elvis]$ ls -l /usr
total 132
drwxr-xr-x    2 root     root        40960 Apr 25 06:36 bin
drwxr-xr-x    2 root     root         4096 Jan 24 18:52 dict
drwxr-xr-x    2 root     root         4096 Jan 24 18:52 etc
drwxr-xr-x    4 root     root         4096 Jan 24 18:52 games
drwxr-xr-x  100 root     root         8192 Apr 11 05:55 include
drwxr-xr-x    8 root     root         4096 Mar 31 21:52 kerberos
drwxr-xr-x   90 root     root        40960 Apr 25 06:36 lib
drwxr-xr-x   10 root     root         4096 Apr 11 05:51 libexec
lrwxrwxrwx     1 root       root              14 Sep 13  2002 local ->
../home/local/
drwxr-xr-x    2 root     root         8192 Apr 25 06:36 sbin
drwxr-xr-x  212 root     root         8192 Apr 23 16:30 share
drwxrwxr-x    5 root     pst          4096 Apr 25 08:12 src
lrwxrwxrwx     1 root       root              10 Apr  1 11:07 tmp ->
../var/tmp
drwxr-xr-x    8 root     root         4096 Jan 24 18:52 X11R6
```

El primer comando **ls** simplemente hace una lista del contenido del directorio. El segundo comando **ls -s**, el cual incluye la opción **-s**, presenta también el tamaño del contenido también. El tercer comando **ls -l** presenta una lista "larga", incluyendo todo tipo de detalles sobre los archivos tales como los permisos, propiedad y la fecha de modificación. Por ahora, no se preocupe de los detalles en la salida, ésto se abarcará en un cuaderno posterior sobre la utilización del sistema de archivos. Sólo observe cómo se utilizan las opciones de la línea de comandos para modificar el comportamiento básico del comando **ls**.

Opciones cortas de la línea de comandos

Observe que las dos opciones que se utilizaron anteriormente **-s** y **-l** son opciones de una sola letra. Estas se conocen como opciones "cortas" de la línea de comandos. A veces estas opciones cortas también pueden tener un argumento. Por ejemplo, el comando **ls** tiene la opción **-w**, la cual especifica la "extensión", de la salida en caracteres. Considere el siguiente ejemplo:

```
[elvis@station elvis]$ ls -w 40 /usr/
bin   games     lib      sbin   tmp
dict  include   libexec  share  X11R6
etc   kerberos  local    src
```

En este caso, la palabra `40` no se considera un argumento del comando **ls** sino que es un argumento de la opción del comando **-w** (¿Qué tan extensa debe ser la salida? 40 caracteres). Los argumentos de las opciones simplemente siguen la opción de la línea de comandos. ¿Cómo se sabe cuales opciones tienen argumentos y cuales no? La respuesta corta es por experiencia, pero pronto mencionaremos formas de encontrar ayuda también.

Múltiples opciones cortas de la línea de comandos

Se puede utilizar más de una opción de la línea de comandos a la vez. Las múltiples opciones simplemente se ponen juntas entre el comando y el argumento. El siguiente ejemplo presenta una nueva opción **-r** para el comando **ls**, el cual invierte el orden de búsqueda. Observe cómo se utiliza junto con las opciones **-s** y **-w**.

```
[elvis@station elvis]$ ls -s -w 40 -r /usr/
total 132
   4 X11R6     0 local        4 games
   0 tmp       4 libexec      4 etc
   4 src      40 lib          4 dict
   8 share     4 kerberos    40 bin
   8 sbin      8 include
```

Con frecuencia, cuando se utilizan múltiples opciones, los usuarios tomarán ventaja del atajo que les permite poner todas las opciones juntas con un sólo guión (`-`), como en el siguiente ejemplo:

```
[elvis@station elvis]$ ls -srw 40  /usr/
total 132
   4 X11R6     0 local        4 games
   0 tmp       4 libexec      4 etc
   4 src      40 lib          4 dict
   8 share     4 kerberos    40 bin
   8 sbin      8 include
```

Todas las opciones con letras individuales que no tienen argumento, en este caso **-s** y **-r**, se pueden poner juntas compartiendo un solo `-`. Si una opción sí tiene argumento tal como **-w 40**, sólo puede compartir un guión si está de última. De esta manera, se puede especificar el argumento justo al lado de la línea de comandos.

Opciones largas de la línea de comandos

En los primeros días de Unix, todas las opciones de la línea de comandos compartían la sintaxis mencionada anteriormente. Con la evolución de Unix, la gente empezó a necesitar lo que se conoce como opciones "largas". A diferencia de las opciones con sólo una letra, las opciones largas están compuestas de palabras. En vez de empezar con un guión inicial, las opciones largas están precedidas de dos guiones (`--`). Algunos comandos utilizan sólo opciones cortas y algunos comandos utilizan opciones largas. Muchos comandos, incluyendo **ls**, manejan ambos.

```
[elvis@station elvis]$ ls --size /usr/
total 132
```

```
 40 bin      4 games       40 lib         8 sbin      0 tmp
  4 dict     8 include      4 libexec     8 share     4 X11R6
  4 etc      4 kerberos     0 local       4 src
```

Cuando las opciones largas tienen un argumento, la sintaxis también difiere un poco. En vez de que el argumento siga la opción como una palabra separada, el argumento se pone junto con la opción larga separada por un =, como en **--width=40**. Observe que las opciones tanto largas como cortas, también se pueden mezclar.

```
[elvis@station elvis]$ ls --width=40 --size -r  /usr/
total 132
   4 X11R6     0 local        4 games
   0 tmp       4 libexec      4 etc
   4 src      40 lib          4 dict
   8 share     4 kerberos    40 bin
   8 sbin      8 include
```

Argumentos

Comparados con las opciones de la línea de comandos, los argumentos son fáciles. Cualquier palabra que quede en la línea de comandos, después del nombre del comando y después de cualquier opción de la línea de comandos se le denomina argumentos del comando. Lo que el comando espera o no como argumento depende del comando. Por ejemplo, si se le da algún argumento al comando **ls** el comando tratará los argumentos como archivos o directorios a enumerar. El comando **ps** no espera ningún argumento. El comando **cal** toma de cero a dos, un mes y un año posibles para poder generar el calendario. El aprender qué argumentos espera un programa y lo que hace con estos es parte del aprender a utilizar ese comando.

Ayuda: uso

¿Cómo recordar todas estas opciones de la línea de comandos? No es necesario. No obstante, los usuarios experimentados de Linux han aprendido a refrescar fácilmente su memoria ya que la mayoría de los comandos soportan la opción **--help** o las opciones cortas **-h** o **-?**. Estas opciones usualmente hacen que el comando emita un mensaje de "uso" en vez de realizar su operación normal. Este mensaje de uso contiene un resumen de los argumentos que se esperan, las opciones que se soportan y lo que hacen. Observe que el mensaje de uso generado por el comando **ls** es bastante largo y ha sido abreviado en la siguiente salida.

```
[elvis@station elvis]$ ls --help
Usage: ls [OPTION]... [FILE]... ❶
List information about the FILEs (the current directory by default).
Sort entries alphabetically if none of -cftuSUX❷ nor --sort.

Mandatory arguments to long options are mandatory for short options
too.
  -a, --all                  do not hide entries starting with .
  -A, --almost-all           do not list implied . and ..
      --author               print the author of each file
  -b, --escape                    print octal escapes for non-graphic
characters
```

```
   ...

  -k                         like --block-size=1K
  -l                         use a long listing format

  ...

  -r, --reverse              reverse order while sorting
  -R, --recursive            list subdirectories recursively
  -s, --size  ❸              print size of each file, in blocks

  ...

  -v                         sort by version
  -w, --width=COLS ❹            assume screen width instead of current
value
  -x                               list entries by lines instead of by
columns

  ...
```

Algunos detalles que se deben observar en el mensaje de uso:

❶ Los elementos opcionales se encuentran entre corchetes (`[ y ]`).

❷ Aquí, el mensaje de uso se refiere a las opciones cortas utilizando la forma abreviada.

❸ Observe que **ls** soporta tanto las formas cortas (**-s**) como largas (**--size**) de esta opción.

❹ Aquí está la opción **-w** o **--width**, las cuales requieren un argumento.

Los mensajes de uso no proporcionan una referencia completa para el comando, sino que simplemente proporcionan información para refrescar la memoria. Más adelante en este cuaderno se discutirán otras formas de encontrar ayuda.

Ejemplos

Aprender a utilizar el comando cat

Una amiga del usuario madonna le dijo que el comando **cat** se utiliza para mirar el contenido de los archivos. Ella nunca ha utilizado este comando antes y está interesada en aprender a utilizarlo. Entonces, empieza por examinar el mensaje de uso del comando.

```
[madonna@station madonna]$ cat --help
Usage: cat [OPTION] [FILE]...
Concatenate FILE(s), or standard input, to standard output.

  -A, --show-all           equivalent to -vET
  -b, --number-nonblank    number nonblank output lines
  -e                       equivalent to -vE
  -E, --show-ends          display $ at end of each line
  -n, --number             number all output lines
```

```
  -s, --squeeze-blank      never more than one single blank line
  -t                       equivalent to -vT
  -T, --show-tabs          display TAB characters as ^I
  -u                       (ignored)
  -v, --show-nonprinting    use ^ and M- notation, except for LFD and
TAB
      --help     display this help and exit
      --version  output version information and exit

With no FILE, or when FILE is -, read standard input.

Report bugs to <bug-coreutils@gnu.org>.
```

Todavía no entiende todo el mensaje de uso, como por ejemplo, las referencias a la entrada y a la salida estándar, pero puede entender lo suficiente de la primera línea como para comprender que el comando **cat** espera los nombres de archivo como su argumento. Intenta mostrar el contenido del archivo `/etc/anacrontab`.

```
[madonna@station madonna]$ cat /etc/anacrontab
# /etc/anacrontab: configuration file for anacron

# See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

1       65      cron.daily              run-parts /etc/cron.daily
7       70      cron.weekly             run-parts /etc/cron.weekly
30      75      cron.monthly            run-parts /etc/cron.monthly
```

Al ver algunas de las opciones de los mensajes de uso observa que la opción **-n** numera las líneas de salida y decide probar esta opción:

```
[madonna@station madonna]$ cat -n /etc/anacrontab
     1  # /etc/anacrontab: configuration file for anacron
     2
     3  # See anacron(8) and anacrontab(5) for details.
     4
     5  SHELL=/bin/sh
     6
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
     7
     8   1          65         cron.daily                    run-parts
/etc/cron.daily
     9   7          70         cron.weekly                   run-parts
/etc/cron.weekly
    10   30         75         cron.monthly                  run-parts
/etc/cron.monthly
```

Desde la salida ahora es fácil ver que este archivo tiene 10 líneas y es fácil referirse a la línea 6. Sospecha que el espacio que hay entre las palabras desde el renglón 8 hasta el 10 son efecto del tabulador en vez del espaciador. Al observar del mensaje de uso que la opción **-t** reemplazará cualquier espacio del tabulador con `^I`, intenta confirmar su sospecha.

```
[madonna@station madonna]$ cat -t /etc/anacrontab
# /etc/anacrontab: configuration file for anacron

# See anacron(8) and anacrontab(5) for details.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

1^I65^Icron.daily^I^Irun-parts /etc/cron.daily
7^I70^Icron.weekly^I^Irun-parts /etc/cron.weekly
30^I75^Icron.monthly^I^Irun-parts /etc/cron.monthly
```

Ahora observa que el uso de la opción **-A** es "equivalente a **-vET**", el cual ella asume que es el atajo de la opción corta **-v**, **-E** y **-T**. Prueba ambos para ver si está en lo correcto.

```
[madonna@station madonna]$ cat -A /etc/anacrontab
# /etc/anacrontab: configuration file for anacron$
$
# See anacron(8) and anacrontab(5) for details.$
$
SHELL=/bin/sh$
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin$
$
1^I65^Icron.daily^I^Irun-parts /etc/cron.daily$
7^I70^Icron.weekly^I^Irun-parts /etc/cron.weekly$
30^I75^Icron.monthly^I^Irun-parts /etc/cron.monthly$
[madonna@station madonna]$ cat -vET /etc/anacrontab
# /etc/anacrontab: configuration file for anacron$
$
# See anacron(8) and anacrontab(5) for details.$
$
SHELL=/bin/sh$
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin$
$
1^I65^Icron.daily^I^Irun-parts /etc/cron.daily$
7^I70^Icron.weekly^I^Irun-parts /etc/cron.weekly$
30^I75^Icron.monthly^I^Irun-parts /etc/cron.monthly$
```

Al ver que las salidas son idénticas, decide que ha interpretado correctamente el mensaje de uso.

Capítulo 5 Administración de terminales

Conceptos clave

- Varias secuencias de control tales como **CTRL-C**, **CTRL-D** y **CTRL-Z** pueden ser útiles para administrar procesos desde la terminal.
- La configuración de una terminal puede ser devuelta a su estado inicial con el comando **reset**.
- La shell **bash** ofrece la oportunidad de completar con el tabulador, la cual le puede ahorrar bastante tiempo al escribir especificando comandos o archivos.
- Se pueden distinguir varios tipos de terminales al saber cómo se refiere Linux a los dispositivos.

Control de terminales

La shell **bash** y el terminal en que ejecuta son las herramientas principales que los usuarios experimentados de Linux utilizan. Más adelante en este curso se dedicará un cuaderno completo a la manera de utilizar las potentes características de la shell de **bash**. Esta lección introduce algunas carcterísticas que las terminales comparten en general y una de ellas que no debería dejarse para más tarde: el completar con el tabulador.

Reinicio de terminales

Cuando las terminales presentan información, muestran un byte a la vez. Cuando los computadores manejan texto humano, los caracteres usualmente se encuentran codificados como un valor de bytes en números enteros utilizando el formato ASCII. Mientras que un byte puede codificar hasta 256 valores diferentes, sólo se utiliza un pequeño número de éstos (alrededor de 100) para caracteres de texto y puntuación. Los otros se utilizan para codificar secuencias de control (vea a continuación) y otras condiciones especiales.

A veces los usuarios accidentalmente (o intencionalmente) le piden a una terminal que presente datos no codificados como texto ASCII, como por ejemplo, una imagen o un ejecutable. La terminal hace lo mejor que puede, pero con frecuencia presenta lo que parecen ser simplemente garabatos. Incluso la terminal puede terminar en un modo alterno de gráficos para que inclusive se vea mal lo que se debe ver como texto ASCII normal.

En estas situaciones, el comando **reset** puede restablecer la terminal a una conducta cuerda. Usualmente, cuando quiere utilizar este comando ni siquiera podrá ver al escribirlo. Sin embargo, si tiene un intérprete de comandos **bash** con caracteres incomprensibles, al escribir **reset** seguido de la tecla **INTRO** usualmente la terminal se restaurará a una conducta cuerda.

Secuencias de control de la terminal

Los terminales Linux comparten bastante con sus ancestros primitivos, los teletipos y las consolas "tontas" o las similares a vt100. Estos dispositivos tenían mecanismos para enviar señales "fuera de banda" o secuencias que señalaban un evento fuera del flujo normal de caracteres escritos tales como un espacio, una campanilla audible o un final de transmisión. Las terminales de Linux, como sus predecesores, utilizan la tecla **CTRL** para enviar esas señales "fuera de banda".

La siguiente tabla resume muchas de las secuencias de control utilizadas con frecuencia que todos los terminales Linux comparten y su uso común en orden alfabético. Después de la tabla se discuten estas secuencias de acuerdo con su utilidad en general.

Table 1. Secuencias de control de la terminal Linux

Combinación de teclas	Nombre simbólico	Uso
CTRL-C	SIGINT	Interrupción anormal - termina el proceso en el primer plano
CTRL-D	EOT	Señalamiento normal del final de la entrada
CTRL-G	BEL	Hacer sonar una campana audible en la terminal
CTRL-H	BS	Tecla de retroceso- borra el caracter anterior
CTRL-J	LF	Introducir una línea - es una alternativa para la tecla **RETURN**
CTRL-L	FF	Form feed-Hace que **bash** limpie la pantalla y que otros programas basados en la pantalla "refresquen" la pantalla actual.
CTRL-Q		Descongelar la pantalla (vea **CTRL-S**)
CTRL-S		Congelar la terminal (descongelar con **CTRL-Q**)
CTRL-U	NAK	Borrar la línea actual
CTRL-Z	SIGSTOP	Suspender el proceso en el primer plano

CTRL-C

CTRL-C es probablemente una de las secuencias más útiles mencionadas anteriormente. Ésta llega al rescate cada vez que un usuario siente que "SOLO QUIERO QUE SE DETENGA". Al utilizar la shell **bash**, escribiendo **CTRL-C**terminará el proceso que se está ejecutando en ese momento y lo devolverá al intérprete de comandos **bash**.

El nodo de dispositivo `/dev/zero` es un pseudo dispositivo que cuando se lee devuelve un número infinito de ceros (binarios). El nodo del dispositivo `/dev/null` es un pseudo dispositivo que bota cualquier información escrita en él. Por lo tanto, el siguiente comando ejecutaría por siempre leyendo y botando ceros. Alguien atascado en esta situación podría utilizar **CTRL-C** para cancelar el comando.

```
[elvis@station elvis]$ cat /dev/zero > /dev/null
(... user types CTRL-C ...)
[elvis@station elvis]$
```

CTRL-D

Como lo verá en un cuaderno posterior muchos de los comandos Unix leen su entrada directamente del teclado. Infortunadamente, para los nuevos usuarios Unix, no siempre es obvio el cómo decirle al comando que pare de escuchar. La respuesta es **CTRL-D**. Esta secuencia se utiliza para enviar un mensaje de "Fin de la Transmisión" (usualmente interpretado como "Final del Archivo") al proceso de escucha.

El comando **wc** es un ejemplo de un comando que al no recibir el nombre de un archivo como argumento entonces contará el número de líneas, palabras y

caracteres digitadas por el usuario. El usuario señala el final de la entrada con **CTRL-D**.

```
[elvis@station elvis]$ wc
polly wants a cracker
polly wants a cracker
polly wants a cracker
(... user types CTRL-D ...)
      3      12      66
[elvis@station elvis]$
```

El comando **wc** reporta obedientemente que el usuario ha escrito 3 líneas, 12 palabras utilizando 66 caracteres (aunque esto no parece útil por el momento, encontrará más usos para **wc** en un cuaderno posterior). ¿Qué pasaría si el usuario hubiese escrito **CTRL-C** en cambio?

CTRL-Z

CTRL-Z se utiliza para suspender un programa, el cual se puede restablecer posteriormente con el comando **fg** ("foreground", primer plano en inglés). En la mitad de una sesión **ftp**, elvis quiere determinar el directorio actual de trabajo de su shell. Suspende la sesión **ftp** con **CTRL-Z**, ejecuta el comando **pwd** desde la shell y luego restablece la sesión **ftp** con **fg.**

```
[elvis@station elvis]$ ftp ftp.redhat.com
Trying 66.187.232.51...
Connected to ftp.redhat.com (66.187.232.51).
220 Red Hat FTP server ready. All transfers are logged. (FTP)
Name (ftp.redhat.com:elvis): anónimo
331 Please specify the password.
Password:
230 Login successful. Have fun.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
(... el usuario escribe CTRL-Z ...)
[1]+  Stopped                 ftp ftp.redhat.com
[elvis@station elvis]$ pwd
/home/elvis
[elvis@station elvis]$ fg
ftp> ftp.redhat.com
ftp> ls
227 Entering Passive Mode (66,187,232,51,67,212)
150 Here comes the directory listing.
drwxr-xr-x    6 ftp      ftp          4096 May 24 01:33 pub
226 Directory send OK.
...
```

La administración de procesos de esta manera es un tema bastante complicado que se estudiará en detalle en un cuaderno posterior. Por ahora, es suficiente con que sepa que **CTRL-Z** suspende y **fg** reanuda.

CTRL-U

CTRL-U usualmente se utiliza para "limpiar" la línea actual. Si ha cambiado tanto una línea de comandos **bash** y sólo quiere volver a empezar en limpio pulse **CTRL-U**. También, **CTRL-U** es muy útil cuando cometa errores al ingresar una contraseña. Como los caracteres usualmente no aparecen en la pantalla, es difícil saber lo que ha escrito. Sólo pulse **CTRL-U** e inicie de nuevo.

CTRL-H

CTRL-H tiene la misma función que la **TECLA DE RETROCESO**. Sin embargo, en algunas terminales la**TECLA DE RETROCESO** se encuentra mapeada de manera incorrecta y por lo tanto no funciona bien. **CTRL-H** casi siempre funcionará en estas situaciones.

CTRL-L

Tradicionalmente, **CTRL-L** se utilizaba para ordenarle a la impresora de línea que expulsara la página actual y que iniciara en la próxima. En los terminales en línea, este comando tiene la función de restablecer la pantalla. Para muchos programas de pantalla, incluyendo los editores, el comando **vi** y **nano**, si la pantalla se torna alguna vez inentendible debido a mensajes inesperados, **CTRL-L** hará que el programa refresque la pantalla, restableciéndola correctamente.

La shell **bash**, **CTRL-L** hace que **bash** deje en limpio la pantalla, pero preserva la línea de comandos actual.

CTRL-Q

CTRL-Q "descongela" una pantalla (haciendo que se active de nuevo) después de que se ha "congelado" con **CTRL-S**. Aunque sólo es útil después de haber utilizado **CTRL-S**, se introduce primero (como el comando más útil) ya que nadie utiliza intencionalmente **CTRL-S**. Por el contrario, los nuevos usuarios de Unix a veces pulsarán por error **CTRL-S** y no entenderán el porqué su terminal no responde. Si alguna vez su terminal parece muerta, intente presionar **CTRL-Q** y vea si eso la arregla.

CTRL-S

Vea **CTRL-Q** arriba.

CTRL-J

Casi nunca se utiliza, (en su lugar, la gente utiliza la tecla **ENTER**).

CTRL-G

El único uso real es que hace pitar la terminal, impresionando a sus amigos con su basta reserva de conocimiento inútil sobre Unix.

Identificación de terminales

Así como con la mayoría de los otros dispositivos, los programas interactúan con los terminales en un nivel bajo por medio de nodos de dispositivos, que se encuentran dentro del directorio `/dev`. Por ejemplo, la comunicación con la primera consola virtual utiliza el nodo de dispositivo `/dev/tty1`. La mayoría de los procesos mantienen un registro de la terminal en donde empezaron y las sesiones de inicio de los usuarios usualmente se encuentran asociadas con la terminal que se encuentran utilizando. Al identificar las terminales en estas situaciones, la terminal se conoce por medio de su nodo de dispositivo, por ejemplo, `tty1`.

En el siguiente extracto, elvis hace una lista de los usuarios actuales en una máquina con **who** y enumera los procesos ejecutándose en ese momento desde su terminal con **ps**. En ambos casos el terminal es especificado en la columna "TTY".

```
[elvis@station elvis]$ who
root     tty2         Jun 21 10:12
elvis    tty3         Jun 21 16:50
blondie  :0           Jun 21 10:13
blondie  pts/0        Jun 21 16:43 (:0.0)
blondie  pts/1        Jun 21 10:14 (:0.0)
blondie  pts/2        Jun 21 10:31 (:0.0)
blondie  pts/3        Jun 21 10:39 (:0.0)
[elvis@station elvis]$ ps
  PID TTY          TIME CMD
 4384 tty3     00:00:00 bash
 4427 tty3     00:00:00 ps
```

Linux trata muchos dispositivos diferentes como una terminal, incluyendo las consolas virtuales, una línea en serie conectada a terminales VT100, módems, etc. La siguiente tabla lista algunos de los nombres comunes para terminales utilizados convencionalmente para algunos de los dispositivos de terminal comúnes.

Table 1. Nombres de los Dispositivos de Terminal

Nombre	Dispositivo	¿Cómo se utiliza?
tty*n*	Consola Virtual	Se accede a esta con la combinación **CTRL-ALT-F*n***.
ttyS*n*	Dispositivo de puerto serial	Un módem o un terminal similar a VT100 adjunto a un puerto en serie. La terminal ttyS0 en Unix se relaciona con COM1 en DOS, ttyS1 en COM2, etc.
pts/*n*	Pseudo-terminal	Una emulación de terminal, con frecuencia utilizado por terminales en el entorno gráfico X, o por shells originadas en la red tal como con **telnet** o **ssh**. Las pseudo-terminales no pueden estar relacionadas directamente a un dispositivo físico.
:0	Servidor X	El servidor X no es realmente una terminal. Cuando un usuario inicia la sesión utilizando el Administrador de Inicio de sesión en el entorno gráfico X, con frecuencia su terminal es listada como el servidor X mismo.

Completar con el tabulador

Completar con el tabulador no es una característica de las terminales sino de la shell **bash** que usualmente ejecuta dentro de éstas. Se puede ahorrar bastante tiempo y esfuerzo aprendiendo a hacer buen uso del tabulador al escribir los nombres de los comandos y archivos.

Al escribir el nombre de un comando como la primera palabra de una línea de comandos **bash**, comience a escribir el comando que quiera ejecutar, pero en algún momento antes de que termine la palabra, pare y pulse la tecla del **TABULADOR**. Puede pasar una de tres cosas:

1. El resto del comando que empezó a escribir aparecerá mágicamente.
2. Parte del comando que estaba escribiendo aparece mágicamente pero luego **bash** para y le pita.
3. **bash** solo le pita.

La shell **bash** hace lo mejor que puede seleccionando el comando que usted empezó a escribir de la lista de todos los comandos posibles. En el primer caso, sólo hay un comando que comienza con las letras que usted escribió así que **bash** puede descubrir exactamente el comando que usted había empezado y lo termina de escribir por usted. Ha terminado.

En el segundo y tercer caso, más de un comando inicia con las letras que usted escribió. **bash** completó el comando tanto como pudo pero paró y le pitó para dejarle saber que usted aún tiene que escoger. Para poder ver las opciones de los comandos que **bash** ha delimitado, oprima el tabulador dos veces y **bash** hará una lista de todos los comandos que inician con las letras que usted escribió inicialmente. Proporcione lo suficiente del comando para que sea único y oprima el tabulador de nuevo. **bash** terminará de escribir el comando por usted.

Un ejemplo es en orden.

El usuario elvis quiere habilitar el modo Unicode para su terminal. Por ahora no se preocupe de lo que es el modo Unicode o el por qué quiere tenerlo. Sólo necesita saber que el comando que habilita este modo se llama **unicode_start** y el comando que lo desactiva es **unicode_stop**. Empieza a escribir el comando y presiona el tabulador.

```
[elvis@station elvis]$ unic<TAB>
```

bash expande la palabra a **unicode_st** y pita. Para poder ver la lista de opciones disponibles para completar, elvis teclea el tabulador dos veces.

```
[elvis@station elvis]$ unicode_st<TAB><TAB>
unicode_start   unicode_stop
```

bash le devuelve una lista con dos opciones posibles que enmpiezan con **unicode_st**. Al ver que el comando que elvis quiere ejecutar es el único que empieza con **unicode_sta**, elvis escribe una **a** y presiona el tabulador.

```
[elvis@station elvis]$ unicode_sta<TAB>
```

bash completa el comando y pone el cursor para que esté listo para comenzar a escribir cualquier argumento u opciones para el comando.

```
[elvis@station rha030]$ unicode_start
```

bash no sólo completa la primera palabra al compararla con los comandos disponibles sino que también se pueden completar las otras palabras en la línea de comandos con base en el sistema de archivos. Por ejemplo, suponga que madonna quiere examinar el archivo `/etc/prelink.conf`. En vez de tener que escribir el nombre completo del archivo, ella puede escribir la primera parte y después presionar el tabulador.

```
[madonna@station madonna]$ cat /etc/prel<TAB>
```

Debido a que `/etc/prelink.conf` es el único archivo que inica con `/etc/prel`, **bash** puede completar todo el archivo.

```
[madonna@station madonna]$ cat /etc/prelink.conf
# This config file contains a list of directories both with binaries
# and libraries prelink should consider by default.
# If a directory name is prefixed with `-l ', the directory hierarchy
# will be walked as long as filesystem boundaries are not crossed.
# If a directory name is prefixed with `-h ', symbolic links in a
# directory hierarchy are followed.
-l /bin
-l /usr/bin
...
```

Al avanzar en el curso se dará cuenta de que el tomarse un poco de tiempo explorando la característica de completar con el tabulador le ahorrará mucho tiempo en adelante. Después de un tiempo, el utilizar el tabulador se convertirá en parte de su naturaleza. Al final de un día muy ocupado, le llegará a doler el dedo meñique izquierdo de tanto presionar el tabulador.

Ejemplos

Reiniciar el terminal después de ver un archivo binario

El usuario blondie accidentalmente utiliza el comando **cat** para ver un archivo ejecutable, **/bin/arch**. Como el archivo es un ejecutable compilado, contiene bytes que no se pueden mostrar en una terminal y esto la descontrola.

```
[blondie@station blondie]$ cat /bin/arch
ELF??414   (44???4????/lib/ld-linux.so.2GNU???y??6,?????aaa????aac????-
?????[???
```

Para restablecer la terminal, **blondie** escribe el comando **reset** a ciegas y su terminal se restablece.

```
[blondie@station blondie]$
```

Cancelar un comando

El usuario prince piensa que puede ser interesante el listar recursivamente el contenido de su sistema de archivos, iniciando con el directorio raíz `/`.

```
[prince@station prince]$ ls -R /
/:
bin   data  etc   initrd  lost+found  mnt  proc  root  sbin  usr  web
boot  dev   home  lib     misc        opt  rha   RPMS  tmp   var

/bin:
arch               cut                gawk        mail               rm
touch
ash            date           gettext   mkdir          rmdir      true
ash.static       dd                 grep        mknod              rpm
umount
aumix-minimal   df                  gtar        mktemp             rvi
uname
awk               dmesg             gunzip     more               rview
unicode_start
basename          dnsdomainname   gzip         mount                sed
unicode_stop
bash              doexec           hostname   mt                 setfont
unlink
bash2            domainname      igawk      mv                 setserial
usleep
...
```

Después de ver pasar los archivos por varios minutos, considera que ha visto lo suficiente. Cancela el comando presionando **CTRL-C**. El comando termina y regresa al intérprete de comandos **bash**.

```
[prince@station prince]$
```

Ejercicios en línea

Suspender la salida de una terminal

Lab Exercise

Objetivo: Aprender a administrar la salida de la terminal para aquellos comandos con una ejecución larga.

Tiempo estimado: 5 minutos.

Especificaciones

1. Como en el ejemplo de prince, tome una lista recursiva del directorio root del sistema de archivos utilizando el comando **ls -R /**.
2. Mientras que esté apareciendo la salida, congele su terminal utilizando la secuencia **CTRL-S**.

3. Descongele la terminal utilizando la secuencia **CTRL-Q**. Debería poder alternar el congelamiento y descongelamiento de la salida utilizando estas secuencias.
4. Mientras que la salida aún se encuentre fluyendo, suspenda el proceso con la secuencia **CTRL-Z**.
5. Confirme que ha suspendido el proceso.

Deliverables

Question 1

1. El comando suspendido **ls -R /**

Limpieza

Después de conseguir el resultado deseado, restablezca el proceso suspendido al primer plano con el comando **fg**. Ahora cancélelo con la secuencia **CTRL-C**.

A organizar las medias

Lab Exercise

Objetivo: Aprender lo útil que es terminar la entrada para un comando que lee desde el teclado.

Tiempo estimado: 5 minutos.

Especificaciones

Usted quisiera generar una lista de medias organizadas (alfabéticamente) en el archivo `ordenar_medias.txt` en su directorio inicial.

1. Ejecute el comando **sort > ordenar_medias.txt** en su directorio inicial. El comando no debe terminar de inmediato sino que debe esperar una entrada del teclado.
2. Escriba varios tipos de medias tales como "medias amarillas", "medias sucias", "medias de winnie pooh", etc. Separe cada tipo de medias con una nueva línea utilizando la tecla **INTRO**.
3. Después de enumerar varias medias termine la lista con la secuencia **CTRL-D**. Ahora debe tener un archivo que contenga una lista de medias ordenadas.
4. Salga de la shell para que se grabe su historia.

```
[student@station student]$ sort > ordenar_medias.txt
medias amarillas
medias sucias
medias de winnie pooh
(... pulse CTRL-D ...)
[student@station student]$ cat ordenar_medias.txt
```

```
medias amarillas
medias de winnie pooh
medias sucias
```

Deliverables

Question 1

1. Un archivo en su directorio inicial llamado `ordenar_medias.txt`, el cual contiene una lista de las medias ordenadas y un archivo `.bash_history` que contiene el comando **sort > ordenar_medias.txt**.

Capítulo 6 Getting Help

Conceptos clave

- La mayoría de los comandos proporcionan resúmenes cuando se invocan las opciones **-h**, **-?** o **--help**.
- Puede encontrar información de referencia mucho más amplia en las "páginas del manual", las cuales se pueden ver con el comando **man**.
- Las páginas **man** tienen capítulos y el contenido de un capítulo anterior puede llegar a obscurecer el contenido de un capítulo en desarrollo.
- Los comandos más complicados se describen de manera más completa en las páginas **info**.
- En Red Hat Enterprise Linux, cualquier documentación menos convencional asociada con un paquete específico se puede encontrar en `/usr/share/doc`.
- El Proyecto de documentación de Linux brinda una gran cantidad de documentación relacionada con Linux.
- Los manuales de Red Hat proporcionan documentación específica a la distribución de Red Hat Enterprise Linux.

Getting Help

Unix y Linux, en particular , tienen la tradición de que los manuales y la documentación no se deben mantener en un estante, sino que se deben encontrar en el sistema de manera que sea de fácil acceso para los usuarios. Nadie recuerda todas las opciones para el comando **ls** así que la mayoría de los comandos proporcionan breves resúmenes llamados "usages" cuando se invocan con las opciones apropiadas. La información de referencia más completa que se puede encontrar para la mayoría de los comandos está en las páginas "man" pages e "info". En Red Hat Enterprise Linux, el directorio `/usr/share/doc` contiene menos documentación estructurada específica para un paquete en particular. Por último, también puede encontrar bastantes manuales y tutoriales en el Proyecto de Documentación Linux o en los paquetes de documentación de Red Hat Enterprise Linux.

Obtención de usos con -h, -? y --help

Al avanzar el curso probablemente se dará cuenta de que a menudo Unix toma decisiones de diseño en favor de la brevedad y la eficiencia más que de la transparencia al nombrar los comandos y las opciones. Esto es en particular cierto para los comandos más utilizados tales como **mv**, **ps** y **vi**. Esto hace de Unix una opción muy efectiva para el usuario conocedor, pero con frecuencia a expensas del aprendiz.

Nadie recuerda todas las funcionalidades de cada uno de los comandos, pero los usuarios experimentados de Unix saben cómo encontrar la información en línea de manera rápida. La primera capa de ayuda con frecuencia la proporcionan los comandos mismos, en la forma de "usages" o breves resúmenes de sintaxis que se producen cuando se invoca el comando con las opciones **-h**, **-?** o **--help**. Los usos se abordaron en la lección anterior pero se mencionan otra vez aquí para que todo esté completo.

Las páginas man

Las páginas del manual, con frecuencia abreviado como "páginas man", son la fuente tradicional de referencia e información de los sistemas Unix. Puede ver las páginas de documentación para la mayoría de los comandos, los formatos de archivos, las llamadas de programación y temas en general, utilizando el comando **man**. Por ejemplo, **man ls** genera la documentación para el comando **ls**.

El paginador less

Red Hat Enterprise Linux usa **less** para ver las páginas del manual. Al ver archivos (incluyendo las páginas man) en **less**, se puede navegar utilizando solo letras: **space** se utiliza para ver la siguiente página, **b** se utiliza para ver la página anterior, **q** es para salir. **less** se abordará en más detalle en una lección posterior; sin embargo, la siguiente tabla resume algunos de los comandos de navegación más útiles al ver las páginas man con **less**.

Table 1. Navegación de less básica

Comando	Acción
space	Ver la próxima página
b	Ver la página anterior
q	Salir
/ *text* **RETURN**	Busca la palabra *text*
n	Encuentra la siguiente ocurrencia de la palabra que se buscó anteriormente

Capítulos Man

Las páginas man están organizadas en ocho capítulos estándares como se muestra en las siguientes tablas. Algunas de las páginas comparten nombres idénticos en diferentes capítulos. Por ejemplo, existen páginas tanto para el comando **passwd**, que se encuentra en el capítulo uno sobre los comandos del usuario, como para el archivo `/etc/passwd`, que se encuentra en el capítulo cinco que cubre los formatos de archivos. Infortunadamente, para el usuario que está tratando de encontrar documentación sobre el formato del archivo `/etc/passwd`, **man passwd** sólo presenta la primera página que encuentra, en este caso la entrada del capítulo uno. Para poder ver la página man del capítulo cinco, se debe especificar explícitamente el capítulo como en **man 5 passwd**.

Table 1. Capítulos Man

Capítulo	Audiencia	Tema
1	usuarios estándar	Comandos
2	desarrolladores	Llamadas del sistema
3	desarrolladores	Llamadas de bibliotecas
4	administradores	Archivos de dispositivos
5	usuarios estándar	Formatos de archivos
6	usuarios estándar	??? (ver las preguntas)
7	usuarios estándar	Información general
8	administradores	Comandos del administrador

En Unix, las referencias a las páginas man usualmente incluyen el número del capítulo en paréntesis despúes del número de la página, tal como *passwd(1)* o *passwd(5)*. Cada capítulo tiene una página de introducción llamada *intro* para que el comando **man 5 intro** presenta una introducción al capítulo 5.

Búsquedas de palabras clave y la opción -a

Dos de las opciones más utilizadas con el comando **man** son **-k**, para realizar búsquedas de palabras y **-a**, para ver todas las "páginas relevantes" para un argumento. La usuaria madonna está tratando de hallar información sobre el formato para el archivo `/etc/passwd`. Ya descubrió que **man passwd** sólo presenta la página man para el comando **passwd**, así que utiliza **man -k** para realizar la búsqueda de la palabra clave en *passwd*.

```
[madonna@station madonna]$ man -k passwd
...
chpasswd             (8)  - update password file in batch
gpasswd              (1)  - administer the /etc/group file
hesiod_free_passwd  [hesiod_getpwnam]  (3)    -  Hesiod  functions  for
retrieving passwd information
hesiod_getpwnam        (3)   - Hesiod functions for retrieving passwd
information
hesiod_getpwuid   [hesiod_getpwnam]   (3)     -   Hesiod   functions   for
retrieving passwd
information
```

```
htpasswd                (1)   - Create and update user authentication
files
ldappasswd           (1)  - change the password of an LDAP entry
lppasswd             (1)  - add, change, or delete digest passwords
pam_localuser        (8)  - require users to be listed in /etc/passwd
passwd               (1)  - update a user's authentication tokens(s)
passwd               (5)  - password file ❶
passwd [sslpasswd]   (1ssl)  - compute password hashes
saslpasswd           (8)  - set a user's sasl password
smbpasswd            (5)  - The Samba encrypted password file
smbpasswd            (8)  - change a user's SMB password
...
```

❶ madonna obtuvo mucho más de lo que estaba buscando, pero incluida en la salida se encuentra la referencia al archivo de la contraseña,y el hecho de que la información se encuentra en el capítulo 5 de las páginas man.

Ahora madonna sabe cuál es el capítulo que necesita así que saca la página con **man 5 passwd**. Otra opción podriá haber sido utilizar la opción **-a**, la cual hace que **man**vea todas las páginas relevantes en orden.

Las páginas info

Las páginas man usualmente están diseñadas para proporcionar información de referencia no tutoriales o ayuda en un contexto en general. Los comandos mucho más complicados se encuentran documentados en las páginas "info" con hipervínculos. Las páginas info preceden los días de los navegadores y los enlaces. Dichas páginas se pueden ver ya sea utilizando el comando **info** tradicional o Red Hat Enterprise Linux proporciona un comando similar con una interfaz más fácil llamada **pinfo**. Las páginas info se utilizan principalmente para software desarrollador por the GNU project.

El comando **pinfo**, sin argumentos, realizará una lista de la tabla de contenidos para todas las páginas info instaladas. La navegación básica es similar al paginador **less**. Los enlaces entre las páginas info se pueden cruzar utilizando las cuatro flechas como se señala en la siguiente tabla.

Table 1. Navegación con pinfo

Comando	Acción
SPACE	Página siguiente
b	Página anterior
q	Salir
/*text* **RETURN**	Busca la palabra *text*
FLECHA DERECHA	Seguir el vínculo
FLECHA IZQUIERDA	Atrás
FLECHA ARRIBA	Enlace anterior
FLECHA ABAJO	Siguiente enlace

El directorio /usr/share/doc

Uno de los principios de diseño detrás del software de código abierto con frecuencia se resume en la frase "lanzamiento temprano, lanzamiento frecuente". Al desarrollar software, este se distribuye tan pronto como sea útil así no se haya pulido o esté bien documentado todavía. El permitir a los usuarios utilizar pronto el software, brinda la ventaja de que ellos pueden ayudar a influir en su diseño continuo.

Red Hat Enteprise Linux adopta esta filosofía e incluirá software estable y útil aún si éste no se encuentra formalmente documentado en las páginas man o en la páginas info. Con frecuencia, la documentación sobre cómo utilizar los productos recién desarrollados puede ser un simple archivo de texto llamado `README`, agrupados de manera rápida por el desarrollador. Red Hat Enteprise Linux incluye tal documentación poco estructurada, pero útil dentro del directorio `/usr/share/doc`, organizado por el paquete de Red Hat propietario del software.

Su dimensión variará de acuerdo con el grado de utilidad que tenga la documentación para un paquete dado. Por ejemplo, elvis descubrió que la documentación informal proporcionada por el visor de archivos PostScript **ggv** es muy poca, ya que consta del `ChangeLog` poco estructurado de los desarrolladores. Con un poco de suerte, la falta de `NEWS` es una buena noticia.

```
[elvis@station elvis]$ ls -s /usr/share/doc/ggv-1.99.97/
total 144
 120 ChangeLog    20 COPYING     0 NEWS     4 TODO
```

En contraste, se puede encontrar mucha información sobre la configuración del complicado servicio para compartir archivos, **samba**, bajo su directorio `/usr/share/doc` incluyendo los subdirectorios que contienen documentación en una variedad de formatos.

```
[elvis@station elvis]$ ls /usr/share/doc/samba-2.2.7a/
autofs   LDAP          misc                README              Roadmap
COPYING  libsmbclient  printer-accounting  README.vfs-recycle
WHATSNEW.txt
docs     Manifest      printing            recycle.conf
```

Manual de Documentación de Red Hat

Red Hat Enterprise Linux incluye manuales de documentación desarrollados como parte de un servicio que Red Hat, Inc presta. Los siguientes manuales de documentación se encuentran disponibles en online, en el CD de documentación en formato RPM o en los libros que vienen como parte del paquete.

- Manual de Instalación de x86
- Manual de Inicio
- Manual para personalización
- Manual de referencia
- Manual de Seguridad
- Manual básico de administración del sistema

Estos manuales dan información específica sobre el sistema operativo de Red Hat Enterprise Linux, incluyendo información básica e instrucciones paso a paso para varias tareas.

El proyecto de documentación de Linux

El proyecto de documentación de Linux ha asumido la poco envidiable tarea de documentar todos los desarrollos rápidos asociados con el sistema operativo Linux. La documentación en este sitio sigue los formatos desarrollados en los primeros días de desarrollo de Linux.

Preguntas más frecuentes

> Las FAQs son una compilación de las *Preguntas más frecuentes* sobre un tema en particular tal como FAQ sobre Linux-RAID.

HOWTOs

> HOWTOs brinda instrucciones paso a paso sobre cómo configurar una faceta en particular de Linux tal como el CD-Writing-HOWTO (como-grabar-CDs) o el ETHERNET-HOWTO (como-eternet).

MANUALES

> Los manuales proporcionan un cubrimiento más profundo sobre temas más amplios tales como la administración de sistemas o incluso la programación del módulo del kernel de Linux.

Aunque la documentación no es específica para la distribución Red Hat Enterprise Linux, gran parte de la información es útil y relevante.

yelp: El navegador de ayuda de Gnome

Por último, presentamos yelp, el navegador de ayuda de Gnome, el cual se puede reiniciar seleccionando "Ayuda" del menú de aplicaciones. yelp proporciona la documentación básica para muchas de las aplicaciones gráficas especificas de GNOME, incluyendo Nautilus File Manager y los applets de GNOME.

Figure 1. yelp: El navegador de ayuda de GNOME